基于短序列分组和拼接策略的子序列快速查询算法 *

范纯龙 1, 2，王靖云 1, 2，滕一平 1, 2，丁国辉 1, 2

(1. 沈阳航空航天大学 计算机学院，沈阳 110136; 2. 辽宁省大规模分布式系统实验室，沈阳 110136)

摘　要：子序列查询技术在金融、商业、医疗等领域均有重要应用，但因 DTW（dynamic time warping）等相似性比对算法的时间复杂度较高，子序列长度对检索时间影响很大，限制了数据集上长子序列检索的效率。针对这一问题，提出一种子序列快速查询算法。该算法首先对数据集中特定长度下所有子序列进行分组并标记出代表性子序列；然后在查询时将查询序列切分成定长的小段序列，并用 DTW 算法确定与小段序列相似的代表子序列候选集；最后对候选集进行序列拼接，获取查询结果序列。实验表明新算法效率较典型算法提高约 10 倍。

关键词：序列数据查询；动态时间规整；子序列

中图分类号：TP391　　doi: 10.19734/j.issn.1001-3695.2018.11.0866

Fast subsequence query algorithm based on short sequence grouping and assembling strategy

Fan Chunlong1, 2, Wang Jingyun1, 2, Teng Yiping1, 2, Ding Guohui1, 2

(1. *School of Computer*, *Shenyang Aerospace University*, *Shenyang* 110136, *China*; 2. *Large-scale Distributed System Laboratory in Liaoning Province*, *Shenyang* 110136, *China*)

Abstract: Subsequence query technique has important applications in several fields such as finance, commerce, and healthcare. However, due to the high time complexity of similarity comparison algorithms such as Dynamic Time Warping (DTW) , the length of subsequence has a great influence on the retrieval time, which limits the efficiency of long subsequence retrieval on data sets. To the end, we present a fast subsequence query algorithm based on short sequence grouping and assembling strategy. In this algorithm, we first separate all subsequences, which are in a given length, in the data set into groups and mark out the representative subsequence for each group. Then, during the query processing, the query sequence is cut into small query sequences in a fixed length, and using DTW algorithm we compute the candidate sets of subsequences, of which the representative subsequence has a high similarity with the small query sequences. Finally, we assemble all the sequences in the candidate sets to derive the query result sequences. Experiments show that the efficiency of the new algorithm is about 10 times higher than that of the typical algorithm.

Key words: sequential data query; dynamic time warping; subsequence

0　引言

时间序列数据是指随着时间变化而形成的有序的数据列表[1]，简称时序数据（time series）。时序数据反映了事物或事件随时间变化的状态，序列中的每个时间点都可以用数值或者符号来表示。时间序列数据广泛存在于金融、商业、医药、气象等应用领域中[2]，对其的数据挖掘在股票价格预测、遗传物质分析、气象预测等领域均有较为广泛的应用[3]。序列数据查询是时间序列挖掘问题中基础且重要的问题，本文重点关注在大型时序数据中不同长度子序列的查询情况。

距离度量是序列查询技术的基础。欧氏距离（ED）因其计算简单且高效，是目前最常用的距离计算方法之一[4, 5]。其他主流的距离度量方法有动态时间规整（DTW）[7]、最长公共子序列（LCSS）[8]、实序列编辑距离（EDR）[8]和实补偿编辑距离（ERP）[8]。这些度量方法都具有很强的鲁棒性，支持时序偏移和不等长序列数据的计算。

时间序列表示是提高查询效率的主要方法。时间序列数据普遍存在维度高和数据量大的特点，对于包含 N 条时间序列数据集，每条序列长度为 n，因此需要计算的子序列总数为 N*n(n-1)/2。例如，来自 UCR 时间序列数据集中的 StarLightCurves 基准数据个数为 9 236，每条序列的长度为 1 024，所以该数据集包含的子序列个数为 4.83*10^9，而现实的数据集一般会比此数据大三个数量级，显然，对所有子序列进行相似性比较是不切实际的。为避免数据量过大而导致系统性能下降，需要将数据表示成低维的形式。常见的表示方法包括离散傅里叶变换（DFT）[9]、离散小波变换（DWT）[10]、奇异值分解（SVD）[11]和分段累积近似（PAA）[12]等。这些方法虽然可以保留大部分原序列所携带的特征信息，但不是将效率作为其主要目标，因此不适用于大型数据集。

范围搜索和最近邻搜索[9]是序列查询的主要目标。通过聚类方法寻找出每一类的代表，进而利用代表进行查询。这类方法主要包括最近邻质心分类器[13]和 K-means 聚类。文献[14]使用数据编辑的方法，既保持了原始数据以确保相似性查询准确性，又提高了分类速度。文献[15]介绍了利用 DTW 算法构建聚类，以及通过改进密度峰值的算法以提高聚类性能。然而该方法的目标是聚类，而不是相似性查询，这与目前的工作不符。

目前，最先进的查询算法（online exploration of time series，

收稿日期：2018-11-12；**修回日期：**2019-02-21　**基金项目：**国家自然科学基金资助项目（61303016）

作者简介：范纯龙（1974-），男，辽宁人，教授，主要研究方向为时序数据分析、社会网络计算；王靖云（1993-），女，辽宁人，硕士研究生，主要研究方向为序列数据相似性检索(1377639283@qq.com)；滕一平（1987-），男，讲师，主要研究方向为时空数据检索、数据隐私保护等；丁国辉（1982-），男，副教授，主要研究方向为数据管理、数据集成、数据库模式匹配.

ONEX）[16]同样面临着准确性与查询效率之间难以权衡的问题，尤其是在查询长子序列时，ONEX 算法很难快速返回查询结果。然而某些系统以时间为代价来提高查询的准确率[17]，缺乏实用性。文献[18]和文献[19]利用 DTW 算法将子序列查询转换为向量查询，但是该方法需要提供诸多参数，限制了查询效率[20]。

本文对现有的查询算法进行改进，提出一种快速查询长子序列的方法 MONEX（modify online exploration of time series）。MONEX 算法将候选序列分成子序列，然后根据子序列之间的相似性关系进行编码以支持快速查询子序列。由于对所有不同长度子序列之间相似性关系进行编码是不可行的，所以选取特定长度进行子序列切分，并利用复杂度较低的点对点距离（如欧氏距离）进行相似性编码。在查询阶段，该方法对查询序列进行切分，以确保长子序列的查询速度，然后采用改进的 DTW 算法进行相似性的计算，最后将相似的序列子段连接成完整的序列。

MONEX 算法是对目前最先进的算法 ONEX 进行的改进。该算法与 ONEX 相比，查询效率提高了近 10 倍，准确率提高了 0.03%。虽然两种算法的准确率非常接近，但在查询长子序列时，MONEX 算法的准确率和效率较现有的 ONEX 算法均有明显提升。

1　预备知识

1.1　基本定义

定义 1　时间序列。已知时间序列为 X，$X=(x_1,x_2,\ldots,x_n)$ 表示包含 n 个真实值的时间序列。D 为时间序列数据集，$D=\{X_1,X_2,\ldots,X_N\}$ 表示包含 N 条时间序列的数据集。

然而现实数据集往往具有高维度和数据量大的特点，因此许多现实场景并不是直接对其进行分析，更多的是将时间序列拆分成较短的子序列然后再进行分析。下面介绍一下时间序列子序列的定义。

定义 2　时间序列子序列[16]。假设存在一条时间序列 $X=(x_1,x_2,\ldots,x_n)$，$X_p[i,j]$ 是长度为 i，起始位置为 j 的时间序列子序列，则 $X_p[i,j]=(x_j,x_{i+1},x_{j+i-1})$ 其中 $1\le i\le n$，$0\le j\le n-1$，$1\le p\le N$。

在进行查询时，一般都是根据设定的相似性阈值来判断两条序列是否相似。下面给出相似性阈值的定义。

定义 3　相似性阈值。通过比较时间序列 X 和 Y 的欧氏距离或 DTW 距离是否小于给定的相似性阈值 ST，进而判断两个时间序列是否相似，表示为 $\mathrm{Dist}(X,Y)\le ST$，其中 $\mathrm{Dist}\in\{\mathrm{ED},\mathrm{DTW}\}$。

查询算法都要选择合适的距离度量方法。本文选择利用欧氏距离和 DTW 距离进行度量，但是由于时间列在搜集时序普遍存在噪声的影响，所以需要使用标准化距离。下面分别给出标准化欧氏距离和标准化 DTW 距离的定义。

定义 4　标准化欧氏距离。给定两个序列 X 和 Y，标准化欧氏距离定义为

$$\overline{ED}(X,Y)=\frac{ED(X,Y)}{\sqrt{n}}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-y_i)^2} \tag{1}$$

定义 5　标准化动态时间规整距离。给定两时间序列 X 和 Y，设 $m\le n$，标准化 DTW 距离定义为

$$\overline{DTW}(X,Y)=\frac{\mathrm{DTW}(X,Y)}{2n} \tag{2}$$

1.2　时间序列在线查询空间相似组

MONEX 算法的核心是建立时间序列在线查询空间相似组。首先证明 ED 与 DTW 之间存在三角不等式，从而能够利用 ED 距离构建一个紧凑型的空间；然后在这种紧凑空间上处理基于 DTW 距离进行查询。

时间序列查询空间相似组，简称相似组，是由候选数据集 D 中特定长度的子序列根据它们之间的相似性关系组成的。由于候选数据集 D 数据量巨大，所以将候选序列只按特定长度切分而不是切分成所有可能的子序列。该特定长度根据数据集的大小按一定比例选取，长度记为 length。

定义 6　代表序列。给定查询空间相似组 S，则集合 S 中序列的逐点平均值被定义为 S 的代表，记做，即对于所有的 $X_p[i,j]\in S$，$R_k^i=avgX_p[i,j]$。

定义 7　查询空间相似组。将数据集 D 中长度为 length 的子序列 $X_p[i,j]$ 放入集合 T 中，假设这些属于 T 的子序列被分别放在具有它们各自代表 R_k^i 的相似组里，使所有子序列处在某一个查询空间相似组里，这个组记为 G_k^i。查询空间相似组需要满足以下两个条件：

a）任意子序列 $X_p[i,j]$ 与其代表 R_k^i 的欧氏距离小于或等于相似性阈值 ST 的一半，即 $\mathrm{ED}(X_p[i,j],R_k^i)\le\frac{\mathrm{ST}}{2}$，$\exists i\in[1,n]$，$\forall j\in[1,n-i]$，$\forall p\in[1,n]$。

b）任意子序列 $X_p[i,j]$ 与其代表 R_k^i 的欧氏距离小于或等于 $X_p[i,j]$ 与其他代表 R_k^i 的欧氏距离，即 $\mathrm{ED}(X_p[i,j],R_k^i)\le\mathrm{ED}(X_p[i,j],R_l^i)$，其中 $\exists i\in[1,n]$，$\forall j\in[1,n\text{-}i]$，$\forall p\in[1,n]$，$\forall l\in[1,g]$，g 表示代表的个数。

引理 1　对于相同查询空间相似组 G_k^i 中两个子序列 X 和 Y，其标准化欧氏距离总是小于或等于阈值，即 $\overline{ED}(X,Y)\le ST$，其中 X，$Y\in G_k^i$。

证明　给出同一查询空间相似组中的两个长度相等的序列 $X=(x_i,\ldots x_j)$ 和 $Y=(y_i,\ldots y_j)$，以及这个组的代表 $R=(r_i,\ldots r_j)$。根据定义 7 可知，$\overline{ED}(X,R)\le\frac{ST}{2}$ 且 $\overline{ED}(Y,R)\le\frac{ST}{2}$，带入到定义 4 中可得

$$\sqrt{\sum_{k=i}^{j}(x_k-r_k)^2}\le\frac{ST}{2} \tag{3}$$

$$\sqrt{\sum_{k=i}^{j}(y_k-r_k)^2}\le\frac{ST}{2} \tag{4}$$

想要证明 $\overline{ED}(X,Y)\le ST$，只需证明 $\sqrt{\sum_{k=i}^{j}(x_k-y_k)^2}\le ST$。对式（3）和（4）两边同时平方可知

$$ED^2(X,R)=\sum_{k=i}^{j}(x_k-r_k)^2\le\frac{ST^4}{4} \tag{5}$$

$$ED^2(Y,R)=\sum_{k=i}^{j}(y_k-r_k)^2\le\frac{ST^4}{4} \tag{6}$$

因为 x_k-y_k 可以表示为 $x_k-r_k+r_k-y_k=(x_k-r_k)+(r_k-y_k)$，所以利用柯西—施瓦茨不等式可以得到 $(x_k-y_k)^2\le 2(x_k-r_k)^2+2(r_k-y_k)^2$。将这个结论与定义 4 和式（3）和（4）结合可得

$$\sum_{k=i}^{j}(x_k-y_k)^2\le 2\sum_{k=i}^{j}(x_k-r_k)^2+2\sum_{k=i}^{j}(y_k-r_k)^2=ST^2 \tag{7}$$

因此可以证明 $\overline{ED}(X,Y)\le ST$ 成立。

定义 8　代表空间。给定数据集 D 中，每个查询空间相似组 G_k^i 对应的代表 R_k^i 以及相关子序列 $X_p[i,j]$ 所组成的空间，

称为代表空间，记为 R-space。

1.3 基于 ED-DTW 三角不等式的时间规整检索

本文的时间序列在线查询系统框架是基于 ED 与 DTW 距离之间的三角不等式关系建立的。由这个关系可知，若查询序列 seq 与查询空间相似组的代表序列相似，则这个序列 seq 与该组中所有的子序列都相似，确保了查询能够在压缩后的 R-space 空间上进行，而不是在整个数据集上对查询序列进行比较，从而提高查询效率。换言之，如果查询序列和代表 R_k^i 之间的标准化 DTW 距离小于或等于相似性阈值的一半，那么该相似组中的所有子序列与该查询序列都相似，并且查询序列与相似组中的所有子序列的标准化 DTW 距离都小于或等于相似性阈值。

引理 2 给定 $Y'=(y_1',\dots,y_n')$ 为任意相似组中的任意子序列，$Y=(y_1,\dots,y_n)$ 为该组的代表序列，$X=(x_1,\dots,x_m)$ 为查询序列，则可推出：若存在 $\overline{ED}(Y,Y')\le\frac{ST}{2}$ 且 $\overline{DTW}(X,Y)\le\frac{ST}{2}$ 则 $\overline{\mathrm{DTW}}(X,Y')\le ST$。

证明 由定义 3 和引理 1 可得

$$\mathrm{ED}(Y,Y')=\sqrt{\sum_{i=1}^{n}(y_i-y_i')^2}\le\sqrt{n}\frac{ST}{2}\tag{8}$$

两边平方可得

$$\mathrm{ED}^2(Y,Y')=\sum_{i=1}^{n}(y_i-y_i')^2\le n\frac{ST^2}{4}\tag{9}$$

因为 DTW 距离在求解过程中需要构造矩阵，所以现需构造矩阵 M(X,Y)和 M(X,Y ')。因为在矩阵 M(X,Y)中弯曲路径 P 是从点（1,1）至点（n,n），那么相应的 DTW 权重最多为 $2n\frac{ST}{2}=nST$，所以现在只需要证明在矩阵 M(X,Y ')中从点（1,1）至点（n,n）的弯曲路径的 DTW 权重最多为 2nST。在矩阵 M(X,Y)中，给定弯曲路径 $P=(p_1,p_2,\dots,p_t,\dots,p_T)$，$n\le T\le 2n-1$，$p_1=(1,1)$，$p_T=(n,n)$，$p_t=(i_t,j_t)$。由引理 2 的已知条件可知，$\sqrt{\sum_{t=1}^{T}(x_{i_t}-y_{j_t})^2}\le 2n\frac{ST}{2}=nST$，将公式两边平方可得

$$\sum_{t=1}^{T}(x_{i_t}-y_{j_t})^2\le n^2ST^2\tag{10}$$

因为 x_i-y_i' 可以表示为 $x_i-y_i+y_i-y_i'=(x_i-y_i)+(y_i-y_i')$,所以利用柯西—施瓦茨不等式可以得到 $(x_i-y_i')^2\le 2(x_i-y_i)^2+2(y_i-y_i')^2$，则

$$\sum_{t=1}^{T}(x_{i_t}-y'_{j_t})^2\le 2\sum_{t=1}^{T}(x_{i_t}-y_{j_t})^2+2\sum_{t=1}^{T}(y_{i_t}-y'_{j_t})^2\tag{11}$$

代入式（10）和（9）可得

$$\sum_{t=1}^{T}(x_{i_t}-y'_{j_t})^2\le 2n^2ST^2+\frac{2n^2ST^2}{4}=\frac{5}{2}n^2ST^2\tag{12}$$

因为 $\frac{5}{2}n^2ST^2$ 恒小于等于 $2n^2ST^2$，所以不等式成立，即 $\overline{\mathrm{DTW}}(X,Y')\le ST$ 成立。

2 MONEX 算法

2.1 算法思想

MONEX 算法从构建查询空间相似组，计算代表距离以及连接子序列这三个步骤执行一次完整查询。具体步骤如下：

a) 将候选序列按照长度 length 切分成所有可能的子序列，根据这些子序列之间的相似关系形成查询空间相似组，与此同时计算出每个查询空间相似组的代表并将这些代表存入代表空间 R-space 之中。将查询序列也按长度 length 切分成小段时间序列。

b) 将上述的小段时间序列与每组代表序列做 DTW 查询，得到与每段时间序列子段最相似的代表序列以及对应的查询空间相似组。

c) 将上述步骤所得的查询空间相似组按照对应的小段时间序列的顺序排列，查找这些相似组中的子序列是否可以按顺序拼接成连续的序列，若可以拼接则拼接起来。

2.2 查询空间相似组构建算法

查询空间相似组构建算法是为了寻找长度为 length 的子序列所在的相似组，然后获得每组相应的代表。

a)将数据集 D 中现有时间序列分解成长度为 length 的所有子序列。为去除连续数据所产生的偏差，使用 RANDOMIZE-IN-PLACE 方法对子序列进行随机排序。

b)将重新排序后的子序列集合中的第一个子序列指定为第一组的成员，并将其指定为第一组的代表。重新排序后选择的代表可以确保相似组不受所提供子序列顺序的影响而产生偏差。

c)将余下的子序列与先前的代表进行比较，即计算两者之间的欧氏距离，选择具有最小欧氏距离的一组，再观察其欧氏距离是否满足小于或等于相似性阈值一半的条件。若条件全部符合，则将该子序列放入目前的查询空间相似组中；否则将其放在新的组中，并被指定为新组的代表。然后重复该步骤，直到所有的子序列都放入到对应的查询空间相似组中。

算法 1 查询空间相似组构建算法

```
Input：数据集 D，相似性阈值 ST，特定长度 i。
Output：代表{R},相似组{G}。
Begin
{G}=∅,{R}=∅
随机化子序列
{X}=特定长度 i 下的所有子序列
minSM=0,  mink=0,  members[]=0
for  X_p ∈ X  do
  if (G= ∅ ) then
    G ← G_1
    G_k ← X_p[i, j]     /*子序列长度为 i,起始位置为 j*/
    G_k.members ← [i, j]     /*将组内子序列的下标计入到 members 数组中*/
    R ← X_p[i, j]
  else
    for k=1 to Representative.count do
      minSM=ED(R, X_p[i, j])
      mink=Representative index
      if  ( min SM ≤ ST/2 ) then
        G_mink ← X_p[i, j]
        G_mink.members = [i, j]
        update  R_mink
      else
        G ← G_{K+1}
        R_{k+1} ← X_p[i, j]
End
```

算法的复杂性是衡量算法好坏的重要指标，具有较高复杂度的算法无法适应于对大型数据的处理。由所给数据可知，本文提出的查询空间相似组的复杂度为 $O(nl^2g)$，其中：n 代

表数据集中时间序列的数量；l 代表被分解成的子序列个数；g 代表查询空间相似组个数。

通常情况下，初始的时间序列都具有固定长度，而且在分组过程中还要将这些序列进行切分，所以时间序列长度这个量不能看做是无穷大量。序列个数 n 随着数据集的大小而增减，而且通常情况下，n 比 l 大很多。因此可以将 l 看做关于 n 的常数。此时的复杂度可概括为 O(ng)。

g 的大小可以用概率来论证。假设在某个时刻有 k 个相似组，并且有一条新加入的时间序列 X，那么新的分组有（k+1）种可能，序列 X 或者属于 k 个组之中的一个序列或者是新建立的第（k+1）组中的序列。假设这些事件发生的概率是相等的，那么序列 X 在新的相似组中的概率是 $\frac{1}{k+1}$。那么新建一个相似组的期望是几何分布，值为 $\frac{1}{p}=k$。因此，$\sum_{k=1}^{g}(k+1)=n$，则变量 g 对应的复杂度为 $O(\sqrt{n})$，所以算法整体的复杂度为 $O(n^{3/2})$。

2.3 多相似性参数设置

如 2.2 节所述，本文的代表空间是由特定的阈值和特定长度的子序列构建而成。然而每个分析师对相似程度的要求不尽相同，需要调节阈值 ST 和子序列的特定长度 length 来制定符合它们要求的查询空间相似组。

定义 9　代表距离。在 R-space 中，每两个代表 R_k^i 和 G_k^i 之间的欧氏距离定义为代表距离 D_c，即 $D_{ckl}^i=ED(R_k^i,R_l^i)$。

定义 10　相似空间。SP-Space 是一个概念上的空间，在其对应的规模下，子序列长度为特定长度 length，相似性阈值为 ST。

基于相似性阈值可以为每个特定长度的子序列集合建立相应的相似空间。一般来说，如果 $ST^{'}\geq ST+D_C$，两个相似组就会合并成一个新的相似组。现有两个指标 SThalf 和 STfinal，在长度为 length 的情况下，当计算出的相似组有一半合并成新的相似组时，则阈值记为 SThalf；当全部的相似组合并时，此时阈值记为 STfinal。例如，现有特定长度 i，SThalf=0.5，STfinal=0.78，也就是说，长度为 i 的子序列组成的所有相似组中，当新阈值等于 0.5 时表明相似组有一半合并了，当新阈值等于 0.78 或者更高时则表明全部合并了。为完成长子序列的查询，将候选序列分解成所有长度的子序列并不现实，这里将特定长度 length 设为 10、20、50、80 和 100 五种。根据长度 length 的不同，分别计算出对应的 SThalf 和 STfinal。根据 SThalf 和 STfinal 这两个参数将相似等级划分如下：

a）“严格”等级，ST<SThalf；

b）“中间”等级，SThalf ≤ ST ≤ STfinal；

c）“模糊”等级，ST>STfinal。

根据参数相似等级，以本文使用数据为例，如果分析师要求给出子序列长度为 i 的“严格”等级的相似性阈值范围的推荐，则推荐值在[0,0.6]内。分析师在此区间内选择的任何值都会返回“严格”相似性的结果。

在该区间中选择较低的值则会导致每个查询空间相似组中的子序列有更“严格”的相似性。

算法 2　不同参数阈值求解

Input：相似性阈值 $ST^{'}$，代表距离 D_c。
Output：查询空间相似组 $\{G'\}$。
Begin
　$G'\leftarrow\varnothing$
if　$ST^{'}$ ==ST then
　　$G'=G$
else if　$ST^{'}$ <ST then
　split groups G
else
　if　$ST^{'}<D_c$　then
　　merge pair of groups with this condition
　else if　$ST^{'}\geq D_c$　then
　　if　$ST^{'}-ST\geq D_c$　then
　　　merge groups with this condition
　else if　$ST^{'}<D_c$　then
　　$G'=G$
end

2.4 基于查询空间相似组的查询过程

本文所研究的查询属于用户驱动的查询类型，即分析师通过提供目标查询序列进行查询。这种查询分为两种类型：

a)查询系统会返回与用户提供的查询序列最相似的时间序列，并且长度也与查询序列相同。例如，用户希望在特定的时间段内查询与苹果股票波动类似的股票，这种情况下，查询序列是数据集中存在的序列。

b)用户可以自己设计所需要的股票波动，并查询数据集中该序列的最佳匹配，这样的序列可能不存在于数据集中，而是查询出的最接近的序列。通常情况下，第一种情形的查询较多。

在查询长子序列时，为了提高算法效率，MONEX 算法提出将查询序列切分成等长的查询序列子段，即把长序列分割成若干个短序列进行查询，然后再将查询到的短序列拼接成与原序列长度相等或相近的序列。下面根据算法要求，对时间序列子段进行定义。

定义 11　时间序列子段。将查询序列切分成 m 个长度为的 l 时间序列子段，表示为 $X_q[m]$，其中 $1\leq m\leq\frac{n}{l}$，且 m 为正整数，n 为查询序列长度。

如 1.2 节所介绍，查询空间相似组是基于 ED 所形成的紧凑序列组，查询系统则在相似组上应用时间规整策略进行查询。具体步骤如下：

a）将查询序列按照特定长度 length 切分成查询序列子段，最后小于长度 length 的子段可以忽略不计。

b）在组间进行查找，计算出每一个时间序列子段与每组代表之间的 DTW 距离，记为 dist。选出每一子段对应的最小 dist 按顺序存入 $best_{num}$ 数组中，并获取该组的序号 G_{numk} 和代表序号 R_{numk}。

c）判断第 numk 组中的子序列是否能与 numk+1 组中的子序列拼接起来，找到前后组中能够拼接成查询序列长度的子序列。

d)找出上述拼接起来的序列其每段子序列与该子序列所在组的代表之间的欧氏距离（这个距离在构建相似组时已经计算过，只需通过子序列的标号即可获取），选取欧氏距离最小的一条序列作为最终查询结果。

算法 3　查询处理器

Input：查询序列 Q，特定长度 length，数组 members。
Output：拼接完成的序列{X}。
Begin
{X}= $\varnothing$，m=n/i，numk=0
$X_q=\{Q_1,Q_2\ldots Q_m\}$
/*查询序列按照特定长度 i 切分成查询序列子段*/
for q=1 to m do

$dist_k = DTW(X_q, G_k.R_k)$

/*计算每一个 X_q 分别与每组的代表的 DTW 距离*/

numk=best distance index

$best_{numk} \leftarrow \min\{dist_k\}$　　/*选出每一个 X_q 对应的最小距离*/

$R_{numk} \leftarrow R_k$

$G_{numk} \leftarrow G_k$

end

for k=1 to m do

if ($G_{numk}.members.[i, j+i]$ && $G_{numk+1}.members.[i, j]) == 1$)

/*判断第 numk 组中的 $X_p[i, j]$ 与第 numk+1 组中的 $X_p[i, j]$ 是否能拼接*/

$\{x\} \leftarrow X_p[2i, j]$

end

end

3　实验结果及分析

本文在真实数据集上，从查询时间和准确率两个方面来评估 MONEX 算法的性能，并与已有的两种方法进行比较。a)使用标准的 DTW 距离对数据集中每条子序列与查询序列进行距离计算，然后找出距离最小的子序列，该算法能够确保计算出最佳的查询结果；b)ONEX[19]方法，该方法首先将序列切分成所有可能长度的子序列，然后用聚类算法分别将不同长度的子序列数据降维，最后在降维数据中进行查询，该方法在查询多维数据时有较好的效果。

实验环境为 Visual 2010，使用语言为 C++，实验在 Inter(R) Core i5-3470 3.2 GHz，4 GB 内存的 Windows7 操作系统上实现。本文从最大的公共时间序列集合 UCR time series collection 中选取数据。选取的数据集有 Face(four)、Symbols、MALLAT、ECG5000 和 HandOutlines，数据点个数分别为 8 400、39 800、56 320、70 000 和 1 002 330。为了使查询效果更加准确，需要将每一条序列进行归一化。具体做法为：找出每一条序列中的最小值(min)和最大值(max)，对于任意一条序列 $X=(x_1, x_2, \ldots, x_n)$，将归一化的点 x_i 记为 $\frac{x_i - min}{max - x_i}$。

3.1　查询时间比对

本文利用标准 DTW 和 ONEX 算法与本文 MONEX 算法进行比对。测试将从不同规格候选序列数据集的查询时间和不同长度查询查询序列的查询时间两个方面进行。

在测试不同规格候选序列数据集的查询时间时，需要将查询 20 次的平均响应时间作为查询时间。此外，为了测试查询序列在数据集中和不在数据集中的查询效果，首先从候选数据集中截取 10 条不同长度的子序列，然后再从其他数据集中截取另外 10 条不同长度的子序列，最后将这 20 条子序列作为查询序列集。查询时间的计算共分两个步骤：a)将每条子序列进行 5 次查询，计算其平均时间作为一次查询的时间；b)记录将 20 条不同长度子序列进行查询的时间，并计算出平均时间作为该算法的查询时间。

MONEX 算法的查询时间比标准 DTW 算法快很多，因此在图 1 中纵轴用对数来表示时间，而且时间的单位 ms。如图 1 所示，本文提出的 MONEX 算法查询时间始终优于标准 DTW 查询算法几个数量级，并且比 ONEX 系统有更短的响应时间。可以看出，在数据集较小的情况下，ONEX 与本系统的查询时间不相上下，但随着候选数据集的增大，查询时间的差异变得明显，本系统的平均查询速度比 ONEX 快 10 倍。

在测试查询查询序列的长度对查询效率的影响时，本文从 HandOutlines 数据集中截取 20 条长度由小到大的子序列作为查询序列集，子序列的长度范围在 20~400 间。将每条子序列进行 5 次查询，计算出平均时间作为该查询序列的查询时间。

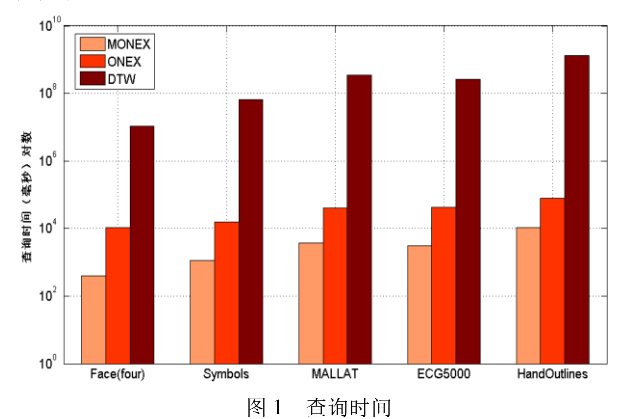


图 1　查询时间

Fig.1　Response time

如图 2 所示，相同的候选数据集下，随着查询序列长度增加，子序列查询时间的差异变得明显，但 MONEX 算法的查询时间始终优于 ONEX 算法，并且查询序列越长，查询速度的优势越明显。

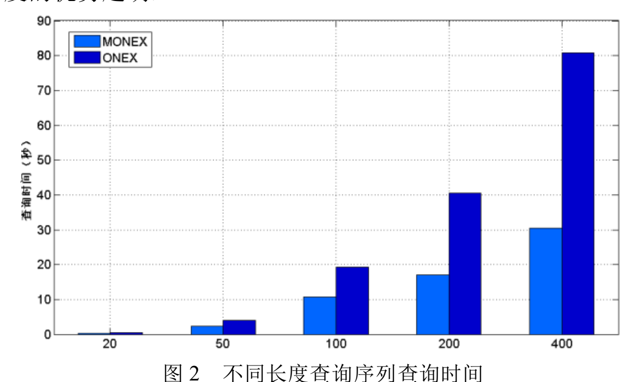


图 2　不同长度查询序列查询时间

Fig. 2　Response time of sample sequences at different lengths

3.2　查询准确率比对

由于 DTW 的运算结果即为两序列间的最小距离，所以使用标准 DTW 距离作为评价标准。由此可知，MONEX 算法准确率的评估即为考察查询所得的子序列与查询序列的距离度量结果对标准 DTW 距离的逼近程度。对此结合文献[21]中对于对比路径准确性的定义以及文献[7]中对查询准确率的定义，将 MONEX 算法准确率的计算定义为

$$Accuracy = [1 - average(err)] * 100\% \qquad (13)$$

$$\text{average}(\text{err}) = \frac{1}{n}\sum_{i}^{n}\frac{apprDist - optimDist}{optimDist} \qquad (14)$$

式（14）中：optimDist 为使用标准 DTW 算法返回的距离；apprDist 为改进查询算法返回的距离。标准 DTW 算法返回的距离 optimDist 已是最相似序列间的距离，而改进的查询算法由于查询偏差将产生更大的距离值，因此近似距离 apprDist 恒大于 optimDist，进而准确率 accuracy 恒大于或等于 0。查询准确率如表 1 所示，不同长度查询序列查询准确率如表 2 所示。

表 1　查询准确率

Table 1　Query accuracy

准确率	Face	symbols	MALLAT	ECG5000	HandOutlines
MONEX	97.36	97.89	97.82	99.21	97.67
ONEX	97.77	97.64	97.78	99.48	97.15

表 2 不同长度查询序列查询准确率

Table 2 Query accuracy of different lengths sequences

准确率	20	50	100	200	400
MONEX	97.06	97.34	97.75	98.03	97.93
ONEX	97.12	97.25	97.18	97.21	97.14

对于上述相似性查询，本节使用与 3.1 节相同的数据集进行查询。由于标准 DTW 距离算法将查询尽可能准确的结果作为目标，所以将它视为标准。由表 1 和 2 可知，本系统的准确率比 ONEX 系统平均高出 0.03%，虽然两者精度接近，但在查询长子序列时 MONEX 系统的准确率有着明显提升。

3.3 预处理评估

在本节中测试本系统在不同相似性阈值下预处理的时间消耗和空间占用情况。图 3 显示了在改变相似性阈值 ST 的情况下，本系统离线构建查询空间相似组的时间。由图 3 可知，对于数值比较低的相似性阈值，其构建时间更长，因为要创建的分组更多。随着相似性阈值的增加，构建时间逐渐变小；当阈值增加到一定程度时，构建的时间将保持不变。也就是说，当阈值增加到一定值时，该长度下的所有子序列都已合并。

在不同相似性阈值下所形成的代表数量可以显示系统在预处理时生成信息的大小。如图 4 所示，相似性阈值越大，代表空间中存储的代表数越少。表 3 显示了在阈值为 0.2 的情况下，各种数据集中代表的个数、切分子序列的个数以及空间占用大小。

表 3 特定数据集下的代表数，子序列数及占用空间

Table 3 Number of representatives under specific data set, number of subsequences and occupied space

数据集	代表数	子序列数	空间占用/ MB
Face(four)	13	7944	1.99
Symbols	24	37900	94.75
MALLAT	128	55275	138.19
ECG5000	67	60500	151.25
HandOutlines	623	995300	2488.25

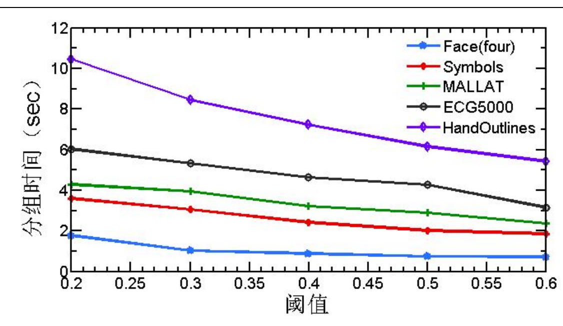


图 3 不同阈值大小下的分组时间

Fig.3 Grouping time in different threshold sizes

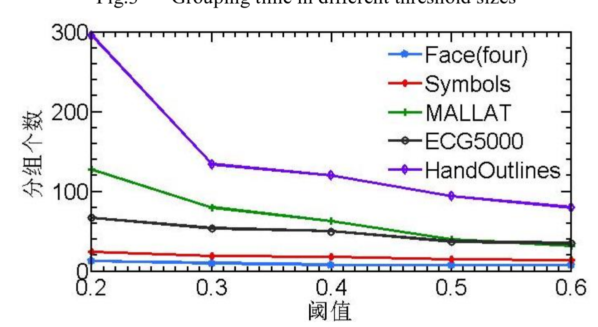


图 4 不同阈值大小下的分组个数

Fig.4 Grouping numbers in different threshold sizes

4 结束语

本文提出了改进的时间序列在线查询算法 MONEX，该方法是基于 ONEX 算法的改进算法。实验结果表明，随着候选数据集的增大，查询时间的差异变得明显，本系统的查询速度比 ONEX 快 10 倍，并且在相同的候选数据集下，随着查询序列长度增大，MONEX 算法在查询时间上的优势也更加明显，与此同时查询的准确率比 ONEX 系统高出 0.03%。

目前查询的最相似子序列长度必须与查询序列保持一致，但在实际查询中，最相似的子序列的长度未必是与查询序列等长的，这可能影响查询的准确率，使查询具有局限性。在今后的工作中，可以改变查询序列的分割策略，使其能够应对不同长度的查询。

参考文献：

[1] Hinich, Melvin J. Time series analysis by state space methods [J]. Technometrics, 2005, 47 (3): 373-373.

[2] Eduardo J R, Vagelis H, Carlos C, *et al.* Correlating financial time series with micro-blogging activity [C]// Proc of the 5th ACM International Conference on Web Search and Data Mining. California: ACM Press, 2012: 513-522.

[3] 范剑青，姚琦伟. 非线性时间序列：建模、预报及应用 [M]. 北京：高等教育出版社, 2005: 1-7. (Fan Jianqing, Yao qiwei. Nonlinear time series: modeling, forecasting and application [M]. Beijing:Higher Education Press, 2005: 1-7.)

[4] Stefan A, Athitsos V, Das G. The move-split-merge metric for time Series [J]. IEEE Trans on Knowledge and Data Engineering, 2013, 25 (6): 1425-1438.

[5] Khatua M, Safavi S H, Cheung N M. Sparse laplacian component analysis for Internet traffic anomalies detection [J]. IEEE Trans on Signal and Information Processing Over Networks, 2018, PP (99): 1-1.

[6] Fu W C, Keogh E, Lau L Y, *et al.* Scaling and time warping in time series querying [J]. Vldb Journal, 2008, 17 (4): 899-921.

[7] Dau H A, Keogh E. Matrix profile V: a generic technique to incorporate domain knowledge into motif discovery [C]// Proc of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2017: 125-134.

[8] Sharif M, Lujo B, Michael K. R. On the suitability of Lp-Norms for creating and preventing adversarial examples [C]// Proc of IEEE CVPRW. Salt Lake City: IEEE Press, 2018: 1686-16888.

[9] Hui Y, Yufang W, Yuan G. Research on similarity mining for flight data [C]// Proc of the 2nd International Workshop on Education Technology and Computer Science. [S. l.] : IEEE Press, 2010: 150-153.

[10] Khandelwal I, Satija U, Adhikari R. Efficient financial time series forecasting model using DWT decomposition [C]// Proc of IEEE CONECCT. [S. l.] : IEEE Press, 2015: 1-5.

[11] Wang Y, Wang P, Pei J, *et al.* A data-adaptive and dynamic segmentation index for whole matching on time series [J]. Very Large Database Endowment, 2013, 6 (10): 793-804.

[12] Cunningham J P, Yu B M. Dimensionality reduction for large-scale neural recordings [J]. Nature Neuroscience, 2014, 17 (11): 1500-9.

[13] Petitjean F, Forestier G, Webb G I, *et al.* Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. [J]. Knowledge and Information Systems, 2016, 47 (1): 1-26, .

[14] Begum N, Ulanova L, Wang J, *et al.* Accelerating dynamic time

warping clustering with a novel admissible pruning strategy [C]// Proc of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [S. l.] : ACM Press, 2015: 49-58.

[15] Neamtu R, Ahsan R, Rundensteiner E, *et al.* Interactive time series exploration powered by the marriage of similarity distances [J]. Very Large Database Endowment, 2016, 10 (3): 169-180.

[16] François P, Forestier G, Webb G I, *et al.* Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm [J]. Knowledge and Information Systems, 2016, 47 (1): 1-26.

[17] Athitsos V, Papapetrou P, Potamias M, *et al.* Approximate embedding-based subsequence matching of time series [C]// Proc of ACM SIGMOD. Athens: ACM Press, 2008: 365-378.

[18] Neamtu R, Ahsan R, Rundensteiner E, *et al.* Interactive time series exploration powered by the marriage of similarity distances [J]. Very Large Database Endowment, 2016, 10 (3): 169-180.

[19] Rakthanmanon T, Campana B, Mueen A, *et al.* Searching and mining trillions of time series subsequences under dynamic time warping [C]// Proc of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. California: ACM Press, 2012: 262-270.

[20] Nagendar G, Jawahar C V. Efficient word image retrieval using fast DTW distance [C]// Proc of the 13th International Conference on Document Analysis and Recognition. Hyderabad: IEEE Press, 2015: 876-880.

[21] Moradi H R, Furuichi S, Minculete N. Estimates for tsallis relative operator entropy [J]. Mathematical Inequalities and Applications, 2017, 20 (4): 1079-1088.